

Quick Reference Guide

Microsoft[®] C Compiler

for the MS-DOS[®] Operating System

© Copyright Microsoft Corporation, 1984, 1985, 1986

Microsoft, the Microsoft logo, MS-DOS, and XENIX are registered trademarks of Microsoft Corporation. CodeView and The High Performance Software are trademarks of Microsoft Corporation.

Microsoft® C Compiler

Quick Reference Guide

Contents

Command-Line Syntax	2
MSC and CL Options	3
Environment Variables	8
Linker Options	9
LIB Command Characters	10
MAKE Summary	11
CodeView™ Options	12
C Escape Sequences	13
C Operators	14
C Type Sizes	16
C Statements	17
Preprocessor Directives	19
Formatting with printf	21
Formatting with scanf	23

Notational Conventions

- **Bold** Bold type indicates text that must be typed exactly as shown.
- *Italics* Italics indicate places where specific terms, supplied by the user, will appear.
- [Brackets] Brackets indicate that the enclosed entry is optional.
- Ellipsis dots ... Horizontal ellipsis dots are used in syntax examples to indicate that additional material may be included but is not shown.

Command-Line Syntax

MSC *sourcefile* [,*objectfile*][,*sourcelistfile*][,*objectlistfile*]]] [*options*] [;]

CL [*options*] *filenames* [-**link** *libraryfield*]

LINK *objectfiles* [,*executablefile*] [,*mapfile*] [,*libraryfiles*]]] [*options*] [;]

LIB *oldlibrary* [/PAGESIZE:*number*] [*commands*] [,*listfile*] [,*newlibrary*]]] [;]

MAKE [*options*] [*macrodefinitions*] *makefilename*

CV [*options*] *executablefile* [*arguments*]

EXEPACK *executablefile* *outputfile*

EXEMOD *executablefile* [*option* [*number*]]

SETENV *filename* [*environmentsize*]

MSC and CL Options

/Aletter

Sets the memory model. The letter may be **S**, **M**, **C**, **L**, or **H**, standing for small, medium, compact, large, and huge model, respectively. The default is **/AS**.

/Astring

Sets the program configuration. The *string* consists of three characters in any order, one from each of the following groups:

Group	Code	Description
Code pointer size	s	Small
	l	Large
Data pointer size	n	Near
	f	Far
	h	Huge
Segment setup	d	SS equal to DS
	u	SS not equal to DS ; DS loaded for each module
	w	SS not equal to DS ; DS fixed

The default is **/Asnd** (equivalent to **/AS**).

/C

Preserves comments when preprocessing a file (use only with **/E**, **/P**, or **/EP**).

/c

CL only. Compiles without linking.

/Didentifier [= [string]]

Defines *identifier* to the preprocessor. The value of *identifier* is *string* if *string* is given. The value is empty if the equal sign is given without *string*. The value is 0 if *identifier* is given without the equal sign.

/E

Preprocesses the source file, copying the result to the standard output and inserting **#line** directives.

/EP

Preprocesses the source file, copying the result to the standard output without **#line** directives.

/F *hexnumber*

CL only. Sets stack size to number of bytes. The default stack size is 2K. This is the same as the **/STACK** option of the linker.

/Fa[*filename*]

Produces an assembly listing.

/Fc[*filename*]

Produces a combined source-assembly listing.

/Fe[*filename*]

CL only. Names the executable file. The default name is *basename.EXE*.

/Fl[*filename*]

Produces an object listing.

/Fm[*filename*]

CL only. Creates a map file.

/Fofilename

Names the object file. The default name is *basename.OBJ*.

/Fs[*filename*]

Produces a source listing.

/FPa

Generates floating-point calls and selects an alternate math library.

/FPc

Generates floating-point calls and selects emulator (uses 8087/80287 if one is present).

/FPc87

Generates floating-point calls and selects 8087/80287 library (requires an 8087 or 80287 at run time).

/FPi

Generates in-line 8087/80287 instructions and selects emulator (uses 8087 or 80287 if one is present). This option is the default.

/FPi87

Generates in-line 8087/80287 instructions and selects 8087/80287 library (requires an 8087 or 80287 at run time).

/G0

Generates 8086/8088 instructions. This option is the default.

/G1

Generates 80186/80188 instructions.

/G2

Generates 80286 instructions.

/Gc

Generates FORTRAN/Pascal call/return sequences and naming conventions for a module.

/Gs

Removes calls to a stack-probe routine.

/Gt[*number*]

Places data items greater than *number* bytes in a new segment. The default is 256 bytes if the option is given with no *number*. The default is 32K if the option is not given. This option is only relevant for compact, large, and huge models.

/Gw

Generates code suitable for use in Microsoft® Windows applications.

/Hnumber

Restricts significant characters of external names to *number* characters. The default is 31 characters.

/HELP

Lists all **MSC** or **CL** options to the standard output. This option is not case sensitive.

/Idirectory

Adds *directory* to the top of the list of directories to be searched for include files.

/J

Changes the default for **char** type from signed to unsigned.

/link libraryfield

CL only. Passes the options or other data in *libraryfield* to **LINK**.

/NDdatasegmentname

Sets the data segment name.

/NMmodulename

Sets the module name.

/NTtextsegmentname

Sets the text segment name.

/Ostring

Controls optimization. The *string* consists of one or more of the following characters:

Code	Description
------	-------------

d	Disables optimization
a	Relaxes alias checking
s	Favors code size
t	Favors execution time
x	Maximizes optimization (equivalent to /Oas /Gs)

The default is **/Ot**.

/P

Preprocesses the source file and sends output to the file with the base name of the source file and the **.I** extension.

/u

Removes definitions of all four predefined identifiers.

/Uidentifier

Removes definition of the given predefined identifier.

/Vstring

Copies *string* to the object file.

/w

Suppresses compiler warning messages.

/Wnumber

Sets the output level (*number* = 0, 1, 2, or 3) for compiler warning messages. The default is 1.

/X

Ignores the list of “standard places” in the search for include files.

/Za

Disables language extensions. These include the keywords **cdecl**, **far**, **fortran**, **huge**, **near**, and **pascal**, and other capabilities.

/Zd

Includes line-number information in the object file.

/Ze

Enables language extensions. These include the keywords **cdecl**, **far**, **fortran**, **huge**, **near**, and **pascal**, and other capabilities. This option is the default.

/Zg

Generates function declarations from function definitions and writes declarations to standard output.

/Zi

Writes to the object file all the symbolic information required by the Microsoft CodeView™ debugger.

/Zl

Removes default library information from the object file.

/Zp

Packs structure members.

/Zs

Performs a syntax check only.

Note CL also accepts most of the options accepted by the Microsoft XENIX® cc command, as described in Appendix C of the *Microsoft C Compiler User's Guide*.

Environment Variables

Variable	Types of Files
PATH	Executable compiler and utility files
INCLUDE	Include files
TMP	Temporary files
LIB	Library files

Note Environment variables, except for **TMP**, can contain multiple paths separated by semicolons.

Linker Options

/CODEVIEW (/CO)

Creates a special-format executable file containing the symbolic information needed by the Microsoft CodeView debugger.

/CPARMAXALLOC: *number* (/CP)

Sets the maximum memory allocation of program to *number*.

/EXEPACK (/E)

Packs the executable file during linking.

/HELP (/HE)

Lists all **LINK** options to standard output.

/LINENUMBERS (/L)

Includes in the listing file the line numbers and addresses of the source statements in the input modules.

/MAP (/M)

Creates a listing file containing all public (global) symbols defined in the input modules.

/NODEFAULTLIBRARYSEARCH (/NOD)

Causes default libraries to be ignored.

/NOIGNORECASE (/NOI)

Causes the linker to distinguish between uppercase and lowercase letters.

/PAUSE (/P)

Causes the linker to pause in the link session so that you can change disks.

/SEGMENTS: *number* (/SE)

Sets the number of segments the linker allows a program to have. The default is 128.

/STACK: *number* (/ST)

Sets the stack size to *number*, which may be any positive value up to 65536 bytes. The default for C programs is 2K (2048 bytes).

Note The linker supports several other options that should never be used with C programs. See the *Microsoft C Compiler User's Guide* if you are linking assembly-language modules or modules from other languages.

LIB Command Characters

Command	Task	Description
+	Add	Appends an object file or library file to the given library
-	Delete	Deletes a module from the library
- +	Replace	Replaces a module by deleting it from the library and appending to the library an object file with the same name
*	Copy	Extracts a module without deleting it from the library and saves the module as an object file with the same name
- *	Move	Extracts a module and deletes it from the library after saving it in an object file with the same name

MAKE Summary

MAKE Options

Option	Action
/D	Displays the last modification date of each file as the file is scanned
/I	Ignores exit codes returned by programs called from the MAKE description file; MAKE continues execution of the next lines of the description file despite the errors
/N	Displays commands that would be executed by a description file, but does not actually execute the commands
/S	Executes in "silent" mode; lines are not displayed as they are executed

Format for Dependency

```

targetfile : dependentfiles [#comment]
[#comment]
command [#comment]
[command] [#comment]
...

```

Format for Declaring Macros

```

name = value

```

Format for Using Previously Defined Macros

```

$(name)

```

Order of Precedence for Macro Names Defined in Several Places

1. Command line
2. Description file
3. Environment variable

Special Macro Names

Name	Value Substituted
\$ *	Base-name portion of the target (without the extension)
\$ @	Complete target name
\$ **	Complete list of dependencies

CodeView™ Options

Option	Action
/B	Starts in black-and-white mode with color/graphics adapter
/C<i>commands</i>	Executes <i>commands</i> on start-up
/F	Starts with screen flipping (exchanges screens by flipping video pages)
/M	Disables the mouse
/T	Starts in sequential mode
/S	Starts with screen swapping (exchanges screens by saving and copying from buffer)
/W	Starts in window mode
/43	Starts in EGA (enhanced graphics adapter) 43-line mode

C Escape Sequences

Sequence	Name
<code>\n</code>	New line
<code>\t</code>	Horizontal tab
<code>\v</code>	Vertical tab
<code>\b</code>	Backspace
<code>\r</code>	Carriage return
<code>\f</code>	Form feed
<code>\a</code>	Bell
<code>\'</code>	Single quotation mark
<code>\"</code>	Double quotation mark
<code>\\</code>	Backslash
<code>\ddd</code>	Character whose ASCII code is octal <i>ddd</i>
<code>\xdd</code>	Character whose ASCII code is hexadecimal <i>dd</i>

C Operators

Precedence	Operator	Name
1	()	Function
	[]	Array
	.	Member selection
	->	Member selection
2	-	Arithmetic negation
	~	Bitwise complement
	!	Logical NOT
	*	Indirection
	&	Address of
	++	Increment
	--	Decrement
	sizeof	Size in bytes
	(type)	Type cast
3	*	Multiplication
	/	Division
	%	Remainder
4	+	Addition
	-	Subtraction
5	<<	Shift left
	>>	Shift right
6	<	Less than
	>	Greater than
	<=	Less than or equal to
	>=	Greater than or equal to

Precedence	Operator	Name
7	<code>==</code>	Equality
	<code>!=</code>	Inequality
8	<code>&</code>	Bitwise AND
9	<code>^</code>	Bitwise exclusive OR
10	<code> </code>	Bitwise inclusive OR
11	<code>&&</code>	Logical AND
12	<code> </code>	Logical OR
13	<code>exp1?exp2:exp3</code>	Conditional
14	<code>=</code>	Simple assignment
	<code>+=</code>	Addition assignment
	<code>-=</code>	Subtraction assignment
	<code>*=</code>	Multiplication assignment
	<code>/=</code>	Division assignment
	<code>%=</code>	Remainder assignment
	<code>>>=</code>	Right-shift assignment
	<code><<=</code>	Left-shift assignment
	<code>&=</code>	Bitwise-AND assignment
	<code> =</code>	Bitwise-inclusive-OR assignment
	<code>^=</code>	Bitwise-exclusive-OR assignment
15	<code>,</code>	Sequential evaluation

C Type Sizes

Type Name	Alternate Name	Storage	Range of Values
char	—	1 byte	– 128 to 127
int	—	Implementation dependent	(– 32768 to 32767 for Microsoft C 4.0)
short int	short	2 bytes	– 32768 to 32767
long int	long	4 bytes	– 2,147,483,648 to 2,147,483,647
unsigned char	—	1 byte	0 to 255
unsigned int	unsigned	Implementation dependent	(0 to 65535 for Microsoft C 4.0)
unsigned short int	unsigned short	2 bytes	0 to 65535
unsigned long int	unsigned long	4 bytes	0 to 4,294,967,295
float	—	4 bytes	Approximately $3.4e - 38$ to $3.4e + 38$ (7-digit precision)
double	long float	8 bytes	Approximately $1.7e - 308$ to $1.7e + 308$ (15-digit precision)

Note Any type size modified by the **unsigned** keyword can be modified by the **signed** keyword instead. The **signed** keyword is useful if the **/J** option has been used to change the default sign of the **char** type.

C Statements

Statement	Action
break;	Terminates execution of the smallest enclosing do , for , switch , or while statement in which it appears.
<pre>{ [decl] ... stmt [stmt] ... }</pre>	Allocates memory for <i>decl</i> and executes each <i>stmt</i> in order of its appearance, except where a <i>stmt</i> transfers control to another location.
continue;	Passes control to the next iteration of the do , for , or while statement in which it appears.
<pre>do stmt while (expr);</pre>	Executes <i>stmt</i> one or more times until <i>expr</i> becomes 0.
[expr];	Evaluates the <i>expr</i> . The <i>expr</i> may be empty.
<pre>for ([init];[cond];[incr]) stmt</pre>	Executes zero or more times. First the <i>init</i> expression is evaluated. Then while the <i>cond</i> evaluates to nonzero, the <i>stmt</i> is executed and the <i>incr</i> expression is evaluated. When the <i>cond</i> becomes 0, control passes to the statement following the for loop.

Statement	Action
goto <i>name</i> ; ... <i>name</i> : <i>stmt</i>	Transfers control directly to the <i>stmt</i> specified by the label <i>name</i> .
if (<i>expr</i>) <i>stmt1</i> [else <i>stmt2</i>]	Executes <i>stmt1</i> if <i>expr</i> is nonzero; otherwise skips <i>stmt1</i> and either executes <i>stmt2</i> if there is an else clause, or passes control to the next statement if there is no else clause.
return [<i>expr</i>];	Terminates execution of the function in which the return statement appears, and returns control and the value of <i>expr</i> to the calling function.
switch (<i>expr</i>) { [<i>decl</i>] ... [case <i>constexpr</i> :] ... [<i>stmt</i>] ... [default : <i>stmt</i>] [case <i>constexpr</i> :] ... [<i>stmt</i>] ... }	Evaluates <i>expr</i> and then executes any <i>stmt</i> associated with the <i>constexpr</i> whose value matches the initial <i>expr</i> . If there is no match with a <i>constexpr</i> , then the statement associated with the default keyword is executed, unless the default keyword is not used. In this case, control passes to the statement following the switch block. Any declarations appearing at the start of the switch block are local to the block.
while (<i>expr</i>) <i>stmt</i>	Executes <i>stmt</i> zero or more times until <i>expr</i> becomes 0.

Preprocessor Directives

Directive	Action
#define <i>ident</i> [(<i>params</i>)] <i>text</i>	Causes all subsequent cases of <i>ident</i> to be replaced with the given <i>text</i> . If <i>params</i> are given after <i>ident</i> , each occurrence of <i>ident</i> (<i>params</i>) is replaced with a version of <i>text</i> modified by substituting actual arguments for formal parameters.
#undef <i>ident</i>	Removes the current definition of <i>ident</i> .
#include " <i>path</i> " #include < <i>path</i> >	Adds the contents of the file specified by <i>path</i> to the current file. If <i>path</i> is enclosed in double quotes, the current directory is searched first. If <i>path</i> is enclosed in angle brackets, the current directory is not searched.
#if <i>expr</i> [<i>text</i>] #elif <i>expr</i> [<i>text</i>] #elif <i>expr</i> [<i>text</i>] ... #else [<i>text</i>] #endif	Checks each <i>expr</i> associated with an #if or #elif directive until a nonzero <i>expr</i> is found, then processes the <i>text</i> associated with that <i>expr</i> . If there is an #else clause, the <i>text</i> associated with it is only processed if no <i>expr</i> in the #if clause is nonzero.

Directive**Action****#line** *const* ["*file*"]

Changes the compiler's internally stored line number to *const*, and changes the internal file name to *file*. The current line number and file name are available through the predefined identifiers `__LINE__` and `__FILE__`.

#pragma checkstack +
#pragma checkstack -
#pragma checkstack

Instructs the compiler to turn off stack probes if the minus sign (-) is given, or turn on stack probes if the plus sign (+) is given. If no sign is given, stack probes are treated according to the default (on unless /Gs was used).

Formatting with printf

Flags

Flag	Description
-	Left justify
+	Prefix positive numbers with a + sign
Space	Prefix positive numbers with a space
#	Prefix hexadecimal or octal numbers with a 0; use a decimal point in real numbers with trailing zeros

Format Types

Type	Description
d	Signed decimal integer
i	Signed decimal integer
u	Unsigned decimal integer
o	Unsigned octal integer
x	Hexadecimal integer with lowercase letters
X	Hexadecimal integer with uppercase letters
f	Floating point
e	Exponential form with lowercase e
E	Exponential form with uppercase E
g	More compact of f or e with lowercase e
G	More compact of f or e with uppercase E
c	Character
s	Null-terminated string
n	Number of characters written so far (produces no output)
p	Far address displayed in hexadecimal

Type Prefixes

Type	Description
h	Short integer for types d , i , o , x , and X
l or L	Long integer for types d , i , o , x , and X
l or L	Double for types e , E , f , g , and G
F	Far for types s and p only
N	Near for types s and p only

Note This is not a complete list. See the *Microsoft C Compiler Run-Time Library Reference* for more information on **printf** formatting.

Formatting with scanf

Flag

Flag	Description
*	Suppress assignment of the next field; the field is scanned but not signed

Format Types

Type	Description	Storage Type
d	Signed decimal	&int
D	Signed decimal	&long
o	Octal	&int
O	Octal	&long
x	Hexadecimal	&int
X	Hexadecimal	&long
i	Decimal, octal, or hex	&int
I	Decimal, octal, or hex	&long
u	Unsigned decimal	&unsigned int
U	Unsigned decimal	&unsigned long
e	Floating point	&float
f	Floating point	&float
c	Character	&char
n	Number of characters read so far	&int
s	String	&char[length + 1]
p	Far address	&(void far *)

Type Prefixes

Type	Description
h	Short integer for types d , i , o , x , and X
l or L	Long integer for types d , i , o , x , and X
l or L	Double for types e , E , f , g , and G
F	Far for types s and p only
N	Near for types s and p only

Note This is not a complete list. See the *Microsoft C Compiler Run-Time Library Reference* for more information on **scanf** formatting.

MICROSOFT®
Microsoft Corporation
16011 NE 36th Way
Box 97017
Redmond, WA 98073-9717

Part No. 048-014-041